

# Oblivious Lookup Tables

Stefan Rass<sup>1</sup>, Peter Schartner<sup>1</sup>, and Markus Wamser<sup>2</sup>

<sup>1</sup> Institute of Applied Informatics, Universität Klagenfurt, Universitätsstrasse 65-67, 9020 Klagenfurt, Austria, {stefan.rass, peter.schartner}@aau.at,

<sup>2</sup> Institute for Security in Information Technology, Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany, wamser@tum.de

**Abstract.** We consider the following question: given a group-homomorphic public-key encryption  $E$ , a ciphertext  $c = E(x, pk)$  hiding a value  $x$  using a key  $pk$ , and a "suitable" description of a function  $f$ , can we evaluate  $E(f(x), pk)$  without decrypting  $c$ ? We call this an *oblivious lookup table* and show the existence of such a primitive. To this end, we describe a concrete construction, discuss its security and relations to other cryptographic primitives, and point out directions of future investigations towards generalizations.

**Keywords.** homomorphic encryption; private function evaluation; secure function evaluation

## 1 Introduction and Concept

This short note is about the following setting: let  $f : X \rightarrow Y$  be a mapping between finite sets. Assume that the sizes of  $X$  and  $Y$  are sufficiently small to permit a specification of  $f$  via a lookup-table. Let  $E(m, \kappa)$  denote a group-homomorphic encryption of a message  $m$  under a key  $\kappa$ , where  $E$  can be symmetric or asymmetric. Let  $E$  be group-homomorphic in the sense that  $E(m_1 \cdot m_2, \kappa) = E(m_1, \kappa) \cdot E(m_2, \kappa)$ , where  $\cdot$  denotes the respective group operations within the plain- and ciphertext space.

In this setting, we consider the following question: given  $E$  and an encrypted value  $c = E(x, \kappa)$ , can we compute  $E(f(x), \kappa)$  without decrypting  $c$ ? We call any such implementation of  $f$  an *oblivious lookup table*, as it shall effectively hide the evaluation of  $f$ , or equivalently, evaluate  $f$  only on ciphertexts by virtue of conventional homomorphic encryption. Becoming more specifically, let  $p = 2q + 1$  be a safe prime (i.e.,  $q$  is a prime too), and let  $\mathbb{G} \subset \mathbb{Z}_p$  denote the  $q$ -order subgroup generated by some element  $g \in \mathbb{Z}_p$ . We first describe the lookup technique in plain form, and subsequently wrap the encryption around the necessary operations. Let  $X = \{x_1, \dots, x_n\} \subseteq \mathbb{G}$  be an enumeration of (distinct) values to be looked up. To each such element  $x_i$ , we associate a vector  $\mathbf{v}_i = (x_i^k)_{k=0}^{n-1} = (1, x_i, x_i^2, \dots, x_i^{n-1})$ . Notice that  $x_i \neq x_j$  whenever  $i \neq j$  implies that the vectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$  are all linearly independent, as they essentially form the rows of a Vandermonde-matrix  $\mathbf{V}$ . Without loss of generality, let us assume  $|X| = n = |Y|$ , say, by allowing multiple occurrences of the same element in  $Y$  in case that  $f$  is not injective. Under this convention, let

the (not necessarily pairwise distinct) elements of  $Y$  be enumerated as  $Y = \{y_1, \dots, y_n\}$ .

We will construct the value of  $f(x_i)$  by a scalar product of  $\mathbf{v}_i$  with a vector-representation of the lookup table. That is, the lookup table itself is a vector  $\ell$  with the property that  $\mathbf{v}_i^T \cdot \ell = f(x_i)$  for all  $i = 1, 2, \dots, n$ . To this end, let us choose an arbitrary but invertible matrix  $\mathbf{U} \in \mathbb{G}^{n \times n}$  with columns  $\mathbf{u}_1, \dots, \mathbf{u}_n$ . Define the lookup table as  $\ell := \mathbf{U} \cdot \alpha$  for some (yet to be determined) vector  $\alpha = (\alpha_1, \dots, \alpha_n)$ . Now, let us look at the scalar product of  $\mathbf{v}_i$  with  $\mathbf{U} \cdot \alpha$  to yield  $f(x_i) \in \mathbb{Z}_p$ . This results in a linear equation  $\alpha_1(\mathbf{v}_i^T \cdot \mathbf{u}_1) + \alpha_2(\mathbf{v}_i^T \cdot \mathbf{u}_2) + \dots + \alpha_n(\mathbf{v}_i^T \cdot \mathbf{u}_n) = f(x_i)$ . Establishing this condition for all  $i = 1, 2, \dots, n$ , we end up observing that, to find  $\alpha$ , we need to solve the linear system  $(\mathbf{V} \cdot \mathbf{U}) \cdot \alpha = (f(x_1), \dots, f(x_n))^T$  for  $\alpha$ . The coefficient matrix  $\mathbf{V} \cdot \mathbf{U}$  is invertible by construction, and hence we can easily lookup values  $f(x_i)$  by computing  $f(x_i) = \mathbf{v}_i^T \cdot \ell$ , taking  $O(n)$  multiplications and additions.

Now, let us see if we can equivalently do all the necessary steps when the pre-image is encrypted. For that matter, we take an element-wise commitment to the  $\mathbf{v}_i$  from above to represent  $x_i$ . That is, the value  $x_i$  now comes committed and encrypted as  $\tilde{E}(x_i, \kappa) := (E(1, \kappa), E(g^{x_i}, \kappa), E(g^{x_i^2}, \kappa), \dots, E(g^{x_i^{n-1}}, \kappa)) = (c_1, \dots, c_n)$ , so that the matrix of exponents remains  $\mathbf{V} = (v_{ij})_{i,j=1}^n$  with  $v_{ij} = x_i^{j-1}$  and as such invertible. Since the order of  $\mathbb{G}$  is a prime, we can – in a setup phase where the exponents are known – straightforwardly work out the values  $\alpha$  and the lookup table  $\ell = (\ell_1, \dots, \ell_n)$ , which is supplied in plain (not encrypted) form to the instance that seeks to evaluate  $f$ .

To evaluate  $f$ , let the encrypted input value  $x_i$  be given as  $\tilde{E}(x_i, \kappa)$ . Then, we can compute the lookup value  $E(f(x_i), \kappa)$  as

$$\begin{aligned} \prod_{k=1}^n c_k^{\ell_k} &= \prod_{k=1}^n E(g^{x_i^{k-1}}, \kappa)^{\ell_k} = \prod_{k=1}^n E(g^{v_{ik}}, \kappa)^{\alpha_1 u_{k1} + \alpha_2 u_{k2} + \dots + \alpha_n u_{kn}} \\ &= \prod_{k=1}^n E(g^{\alpha_1 v_{ik} u_{k1} + \alpha_2 v_{ik} u_{k2} + \dots + \alpha_n v_{ik} u_{kn}}, \kappa) = E(g^{f(x_i)}, \kappa). \end{aligned} \quad (1)$$

The last equality is instantly obtained by writing out the exponents for  $k = 1, 2, \dots, n$  and rearranging terms properly when summing up.

A final remark is judicious here: the formula yields only a single value based on an input vector. To properly implement the lookup to be *repeatable*, i.e., to model iterations like  $f(f(\dots f(x) \dots))$  or generally functions  $f : X \rightarrow X$ , we need to look up all the elements of the output vector via separate tables. So, the overall lookup table is no longer a  $n$ -dimensional vector, but an  $(n \times n)$ -matrix  $\mathbf{L} = (\ell_1, \dots, \ell_n)$ . The  $j$ -th such lookup table  $\ell_j$  must then be designed to return  $y^{j-1}$ , whenever the input value  $x$  is represented by a sequence  $1, x, x^2, \dots, x^{n-1}$  in the exponents. That is, the mapping  $f(x) = y$ , acting on  $x$  being represented by encrypted values  $1, g^x, g^{x^2}, \dots, g^{x^{n-1}}$ , requires  $n$  lookups that successively yield  $1, g^y, g^{y^2}, \dots, g^{y^{n-1}}$ , each of which by (1) requires  $O(n)$  exponentiations and multiplications. So, the total cost of an oblivious lookup comes to  $O(n^2)$  exponentiations (subsuming multiplications as the cheaper operation here).

Considering security, each lookup table is indeed available in plaintext, but since it is independent of a particular input and the input and output values remain encrypted at all times, knowledge of  $\mathbf{L}$  cannot release any information about the secret  $x$  being transferred into the secret result  $f(x)$ . Probabilistic encryptions like ElGamal offer the additional appeal of enforced re-randomization of the resulting ciphertexts. That is, if a distrusted third party does several lookups, it nevertheless cannot recognize any results as being identical to previous ones.

## 2 Related Work

This work closely relates to Private Function Evaluation (PFE), which provides a system where the function-to-be-evaluated  $f$  and the inputs are private and the evaluator learns nothing about either aside from the (encrypted) results of the evaluation of the function on the inputs. This can be realized using Secure Function Evaluation (SFE) over a universal circuit ([4, 7]), to which  $f$  has to be converted first. Another approach is to use a (non-universal) circuit representation of  $f$  and employ a Fully Homomorphic Encryption (FHE) scheme [2, 6]. However, all mentioned approaches carry complexities that are too high for practical applications. Conceptually closest to our ideas seem to be [3] and [5], both based on singly homomorphic encryption. The former realises PFE in a strict two-party setting with one party providing the function and the other providing the inputs. Evaluation is done through a common virtual machine. The latter is based on a framework that splits the task into Circuit Topology Hiding (CTH) and Private Gate Evaluation (PGE) which together enable PFE with linear complexity in all standard settings. However, both PFE protocols require an interactive setting while we are aiming for the non-interactive setting. The security implications tied to our simple scheme when being lifted to two-operand functions (if that is possible at all) are, however, far from clear and probably intricate (cf. [1]) and will be discussed along the research sketched in this abstract.

## 3 Open Issues

A yet open issue is a proper formalization of security for oblivious lookup tables. Intuitively, the attacker should be unable to learn anything meaningful from the lookup table as such, since this is nothing but a bunch of ciphertexts and hence indistinguishable from self-made cryptograms, provided that the encryption is semantically secure. However, a full-fledged formal argument and definition of security is subject of future considerations. Also, the idea does not obviously generalize to functions of multiple inputs, which poses another interesting question for future research.

## References

1. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Proceedings of the 8th conference on*

- Theory of cryptography (TCC)*, TCC'11, pages 253–273, Berlin, Heidelberg, 2011. Springer.
2. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.
  3. Jonathan Katz and Lior Malka. Constant-Round private function evaluation with linear complexity. In DongHoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 556–571. Springer Berlin Heidelberg, 2011.
  4. Vladimir Kolesnikov and Thomas Schneider. A practical universal circuit construction and secure evaluation of private functions. In Gene Tsudik, editor, *Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 83–97. Springer Berlin Heidelberg, 2008.
  5. Payman Mohassel and Saeed Sadeghian. How to hide circuits in MPC an efficient framework for private function evaluation. In Thomas Johansson and PhongQ Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 557–574. Springer Berlin Heidelberg, 2013.
  6. Alice Silverberg. Fully homomorphic encryption for mathematicians. Cryptology ePrint Archive, Report 2013/250, 2013. <http://eprint.iacr.org/>.
  7. Leslie G. Valiant. Universal circuits (preliminary report). In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 196–203, New York, NY, USA, 1976. ACM.

## Acronyms

**CTH** Circuit Topology Hiding  
**FHE** Fully Homomorphic Encryption  
**PFE** Private Function Evaluation  
**PGE** Private Gate Evaluation  
**SFE** Secure Function Evaluation